Temporal flows in Temporal networks

Eleni C. Akrida* Jurek Czyzowicz† Leszek Gąsieniec‡ Łukasz Kuszner§
Paul G. Spirakis¶

Abstract

We introduce temporal flows on temporal networks. We show that one can find the maximum amount of flow that can pass from a source vertex s to a sink vertex t up to a given time in Polynomial time. We provide a static *Time-Extended network* (TEG) of *polynomial size to the input*, and show that temporal flows can be decomposed into flows, each moving through a single s-t temporal path. We then examine the case of unbounded node buffers. We prove that the maximum temporal flow is equal to the value of the minimum *temporal s-t cut*. We partially characterise networks with random edge availabilities that tend to eliminate the s-t temporal flow. We also consider mixed temporal networks, where some edges have specified availabilities and some edges have random availabilities; we define the truncated expectation of the maximum temporal flow and show that it is **#P**-hard to compute it.

Keywords: temporal networks, network flows, random input, edge availability.

1 Introduction and motivation

1.1 Our model and the problem

It is generally accepted to describe a network topology using a graph, whose vertices represent the communicating entities and edges correspond to the communication opportunities between them. Consider a directed graph (network) $G(V, E)$ with a set V of n vertices (nodes) and a set E of m edges (links). Let $s, t \in V$ be two special vertices called the *source* and the *sink*, respectively; for simplicity, assume that no edge enters the source s and no edge leaves the sink t. We also assume that a very large amount of a quantity, say, a liquid, is available in s at time zero. However, our network is *ephemeral*; each edge is available for use only at certain *days* in time, described by positive integers, and after some (finite) day in time, no edge becomes available again. For example, some edge $e = (u, v)$ may exist only at days 5 and 8; the reader may think of these days as instances of availability of that edge. Our liquid, located initially at node s, can flow in this ephemeral network through edges only at days at which the edges are available.

Each edge $e \in E$ in the network is also equipped with a *capacity* $c_e > 0$ which is a positive integer, unless otherwise specified; the capacities of the edges remain constant over time. We also consider each node $v \in V$ to have an internal buffer (storage) $B(v)$ of maximum size B_v; here, B_v is also a positive integer; initially, we shall consider both the case where $B_v = +\infty$, for all $v \in V$, and the case where all nodes have finite buffers. From Section 4.1 on, we only consider unbounded (infinite) buffers.

The *semantics* of the flow of our liquid within G are the following:

- Let an amount x_v of liquid be at node v, i.e., in $B(v)$, at the *beginning* of day l, for some $l \in \mathbb{N} = \{1, 2, \ldots\}$. Let $e = (v, w)$ be an edge that exists at day l. Then, v may *push* some of the amount x_v through e at day l, as long as that amount is at most c_e. This quantity will arrive to w at the *end* of *the same day*, l, and will be stored in $B(w)$.

*University of Liverpool, Department of Computer Science,Ashton Building, Ashton Street, Liverpool L69 3BX, UK. Email: e.akrida@liverpool.ac.uk

†Université du Québec en Outaouais, Dépt. d'informatique, Gatineau, QC, Canada. Email: jurek@uqo.ca

‡University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool L69 3BX, UK. Email: L.A.Gasieniec@liverpool.ac.uk

§University of Gdańsk, Faculty of Mathematics, Physics and Informatics, Institute of Informatics, ul. Wita Stwosza 57, 80-952 Gdańsk, Poland. Email: lkuszner@inf.ug.edu.pl

¶University of Liverpool, Department of Computer Science, Ashton Building, Ashton Street, Liverpool L69 3BX, UK and University of Patras, School of Engineering, Department of Computer Engineering & Informatics, GR 265 00, Patras, Greece. Email: p.spirakis@liverpool.ac.uk

- At the end of day l, for any node w, some flows may arrive from edges (v, w) that were available at day l. Since each such quantity of liquid has to be stored in w, the sum of all flows incoming to w plus the amount of liquid that is already in w at the end of day l, after w has sent any flow out of it at the beginning of day l, must not exceed B_w.

- Flow arriving at w at (the end of) day l can leave w only via edges existing at days $l' > l$.

Thus, our flows are not flow rates, but flow amounts (similar to considerations in *transshipment problems*). Also, those amounts of flow proceed "in parallel" via edges that exist at the same day, provided that they were located at the starting points of those edges at the beginning of that day.

It is worth mentioning here that using "days" to describe edge availability is not standard in flow literature; this is because the vast majority of the existing literature on flows considers static networks. However, with the availability of the network's edges now depending on (discrete) time, it is only natural to view the time points of availability of an edge as *days* of availability. This terminology is often used in temporal networks literature, and also allows for an easier understanding of flow movement through an edge at a particular time l of availability of that edge (flow leaves one endpoint of the edge at the beginning of the day, traverses the edge during the day, and will have arrived at the other endpoint by the end of the day).

Notice that we assume above that we have absolute knowledge of the days of existence of each edge. This information is detailed, but it can model a range of scenarios where a network is operated by many users and detailed description of link existence (or lack thereof) is needed; for example, one may need to have detailed information on planned maintenance on pipe-sections in a water network to assure restoration of the network services, and one may need to know in advance the time schedule of a rail network to circulate passengers. However, such a detailed input can not be used in all practical cases; often, instead of having a specific list of days of existence of some edge(s), one may be able to obtain statistical knowledge of a pattern of existence of connections via previously gathered information. A model that captures such cases is the model of *Mixed Temporal Networks*, which we introduce and study here, along with the traditional Temporal Networks model.

We provide efficient solutions to the *Maximum temporal flow problem* (MTF): Given a directed graph G with edge availabilities, distinguished nodes s, t, edge capacities and node buffers as previously described, and also given a specific day $l' > 0$, find the maximum value of the quantity of liquid that can arrive to t by (the end of) day l'.

Notice that no flow will arrive to t in fewer days than the "temporal distance of t from s" (the smallest *arrival time* of any $s \to t$ path with strictly increasing days of availability on its consecutive edges; here, *arrival time* is the day of availability of the last edge on the path).

1.2 Previous work and relation to our model

(a) Work on temporal networks Temporal networks were first defined by Kempe et al. [36], and are graphs *the edges of which exist only at certain instants of time, called labels* (see also [42]). So, they are a type of *dynamic* networks. For a recent attempt to integrate existing models, concepts, and results on dynamic networks, see the survey papers by Casteigts et al. [15–17]. Various aspects of temporal (and other dynamic) networks were also considered in the work of Erlebach et al. [22] and in [4–6, 9, 30, 43–45, 52]; as far as we know, this is the first work to examine flows on temporal networks. Berman [13] proposed a similar model to temporal networks, called *scheduled networks*, in which each edge has separate departure and arrival times; he showed that the max-flow min-cut theorem holds in scheduled networks, when edges have *unit capacities*. There is also literature on models of temporal networks with random edge availabilities [3, 18, 19], but to the best of our knowledge, ours is the first work on flows in such temporal networks.

(b) Work on static network flows and transshipment problems Traditional (static) network flows were extensively studied in the seminal book of Ford and Fulkerson [24] (see also Ahuja et al. [2]) and the relevant literature is vast. They have recently been re-examined for the purpose of approximating their maximum value or improving their time complexity [1, 7, 10, 11, 21, 28, 40, 41, 46, 50, 51, 53], and have also been used in multi-line addressing [20].

Network flows are also closely related to *transshipment problems*. In a transshipment problem, shipments of products (i.e., of amounts of products, in analogy to amounts of flows in our model) are allowed between source-sink pairs in a network, where each source has some supply and each sink has some demand. In some applications, shipments may also be allowed between sources and between sinks. Transshipment problems have also been extensively studied in literature; for example, in studies on the *quickest* transshipment problem [31, 34], the authors consider networks with transit times on their edges and study the problem of sending exactly the right amount of flow out of each source and into each sink in the minimum overall time. Other authors have considered problems such as minimising capacity violations in transshipment networks [49], where the initial capacity constraints render the problem infeasible, but an increase in the capacities by some additive terms (the *capacity violations*) allow a feasible shipment so as to minimise an objective function.

(c) Work on dynamic flows Similarly to the above models, *dynamic network flows* (see, e.g., [33]) refer to *static* directed networks, the edges of which have capacities as well as transit times. Ford and Fulkerson [24] formulated and solved the dynamic maximum flow problem. For excellent surveys on dynamic network flows, the reader is also referred to the work of Aronson [8], the work of Powell [48], and the great survey by Skutella [54]. Dynamic network flows are also called *flows over time*. In [23], the authors review *continuous* flows over time where $f_e(\theta)$ is the rate of flow (per time unit) entering edge e at time θ; the values of $f_e(\theta)$ are assumed to be Lebesgue-measurable functions. In the model we consider here, we assume that any flow amount that can pass through an edge at an instant of existence, will pass, i.e., our $f_e(\theta)$ is infinite in a sense. In a technical report [29], the authors examine earliest arrival flows with time-dependent travel times and edge capacities; they describe the flow equations of their model and give their own Ford-Fulkerson approach and dynamic cut definitions; although different to their model, our work gives an intuitively simpler definition of a temporal cut. For various problems on flows over time, see [12, 23, 26, 27, 32, 35, 38, 39]. Flows over time have been also considered in problems of scheduling jobs in a network [14].

(d) Comparison to our model Perhaps the closest model in the exisiting flows literature to the one we consider is the "*Dynamic*[1] dynamic network flows", studied by Hoppe in his PhD thesis [33, Chapter 8]. In [33, Chapter 8], Hoppe introduces *mortal edges* that exist between a start and an end time; still, Hoppe assumes transmission rates on the edges and the ability to hold any amount of flow on a node (infinite node buffers). Thus, our model is an extreme case of the latter, as we assume that edges exist only at specific days (instants) and that our transit rates are virtually unbounded, since at one instant *any amount* of flow can be sent through an edge if the capacity allows.

Notice that the problem that we introduce and study here, MTF, is also different from both the standard maximum flow problem and the transshipment problem. Indeed, in the network of Figure 1 with all node buffers and edge capacities being infinite, but *all edges existing only at the same day*, say $l = 5$, no flow can *ever* arrive to t.

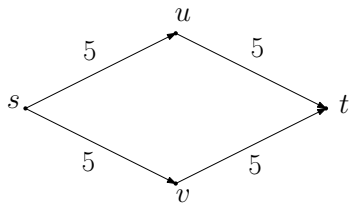


Figure 1: Difference between temporal flows and standard flows.

Also, in our model, the existence of node buffers (holdover flow) is *necessary*; *in contrast to all previous flow and transshipment studies*, our networks cannot propagate flow without holdover flows, i.e., node buffers storing flow units.

[1]The first "dynamic" term refers to the dynamic nature of the underlying graph, i.e., appearance and disappearance of its edges

1.3 Our results

We introduce flows in Temporal Networks for the first time. We consider ephemeral networks that change over time, as well as flows that are dynamic and the movement of which is determined by the temporal structure of the network. We are interested in the maximum total amount of flow that can pass from s to t during the lifetime of the network; notice that the edges of the network exist only at some days during the lifetime, different in general for each edge.

In Section 1.4, we formulate the problem of computing the maximum temporal flow, *MTF*. In Section 2, we show by means of an LP formulation that MTF can be solved in polynomial time, even when the node capacities are finite and possibly different for every node. Note that an NP-hardness result was conjectured by Hoppe [33, personal communication with Klinz] for bounded holdover flows in *dynamic dynamic* networks, which is the model closest to ours. Our result (that MTF is in P) is due to the fact that our model is an extreme case of the dynamic dynamic network flows that avoids the computational hardness.

In Section 3, we define the corresponding *time-extended network* (TEG) which converts our problem to a static flow problem (similarly to the time-extended network tradition in the literature [24]). However, our time-extended network is simplified so that its size, i.e., number of nodes and edges, is *polynomial on the input*, and not exponential as usual in flows over time. We also show that temporal flows are always decomposable into a set of flows, each moving through a particular journey, i.e., directed path whose time existence of successive edges strictly increases.

The remainder of the paper mainly concerns networks in which the nodes have unbounded buffers, i.e., buffers with infinite capacity. In Section 4.1, using the TEG, we prove our *maximum temporal flow-minimum temporal cut* theorem; temporal cuts extend the traditional cut notion, since the edges included in a cut need not exist at the same day(s) in time.

Admittedly, the encoding of the input in our temporal network problems is quite detailed but as previously mentioned, specific description of the edge availabilities may be required in a range of network infrastructure settings where there is a planned schedule of link existence. On the positive side, some problems that are weakly NP-hard in similar dynamic flow models become polynomially solvable in our model. However, in many practical scenarios it is reasonable to assume that not all edge availabilities are known in advance, e.g., in a water network where there may be unplanned disruptions at one or more pipe sections; in these cases, one may have statistical information on the pattern of link availabilities. In Section 4.2.1, we demonstrate cases of temporal flow networks with randomly chosen edge availabilities that eliminate the flow that arrives at t asymptotically almost surely. We also introduce and study flows in mixed temporal networks for the first time; these are networks in which the availabilities of some edges are random and the availabilities of some other edges are specified. In such networks, the value of the maximum temporal flow is a random variable. Consider, for example, the temporal flow network of Figure 2 where there are n directed disjoint two-edge paths from s to t. Assume that *every* edge independently selects a *unique* label uniformly at random from the set $\{1, \ldots, \alpha\}$, $\alpha \in \mathbb{N}$. The edge capacities are the numbers drawn in the boxes, with $w'_i \geq w_i$ for all i. Here, the value of the maximum $s \to t$ flow is a random variable that is the sum of Bernoulli random variables. This already indicates that the exact calculation of the maximum flow in mixed networks is a hard problem. In Section 4.2.2 we show for mixed networks that it is **#P**-hard to compute tails and expectations of the maximum *truncated* temporal flow, to be precisely defined therein.

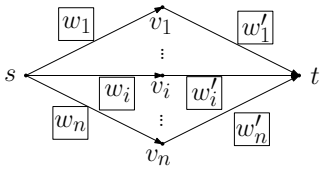


Figure 2: A special case of a mixed temporal network, where no edge has specified availabilities.

We note here that in figures of (temporal) networks shown throughout the paper, a number written in a box next to an edge indicates the edge's capacity and numbers written (with no box) next to the edge indicate the edge's labels.

1.4 Formal Definitions

Definition 1 ((Directed) Temporal Graph)**.** *Let $G = (V, E)$ be a directed graph. A (directed) temporal graph on G is an ordered triple $G(L) = (V, E, L)$, where $L = \{L_e \subseteq \mathbb{N} : e \in E\}$ assigns a* finite *set L_e of discrete labels to every edge (arc) e of G. L is called the* labelling *of G. The labels, L_e, of an edge $e \in E$ are the* integer time instances (e.g., days) *at which e is available.*

Definition 2 (Time edge)**.** *Let $e = (u, v)$ be an edge of the underlying digraph of a temporal graph and consider a label $l \in L_e$. The ordered triplet (u, v, l), also denoted as (e, l), is called* time edge*. We denote the set of time edges of a temporal graph $G(L)$ by E_L.*

A basic assumption that we follow here is that when a (flow) entity passes through an available edge e at time t, then it can pass through a subsequent edge only at some time $t' \geq t+1$ and only at a time at which that edge is available. In the tradition of assigning "transit times" in the dynamic flows literature, one may think that any edge e of the graph has some *transit time*, tt_e, with $0 < tt_e < 1$, but *otherwise arbitrary and not specified*. Henceforth, we will use $tt_e = 0.5$ for all edges e, without loss of generality in our results; any value of tt_e between 0 and 1 will lead to the same results in our paper.

Definition 3 (Journey)**.** *A* journey *from a vertex u to a vertex v, denoted as $u \to v$* journey*, is a sequence of time edges (u, u_1, l_1), (u_1, u_2, l_2), $\ldots$, (u_{k-1}, v, l_k), such that $l_i < l_{i+1}$, for each $1 \leq i \leq k-1$. The last time label, l_k, is called the* arrival time *of the journey.*

Definition 4 (Foremost journey)**.** *A $u \to v$ journey in a temporal graph is called* foremost journey *if its arrival time is the minimum arrival time of all $u \to v$ journeys' arrival times, under the labels assigned to the underlying graph's edges. We call this arrival time the* temporal distance*, $\delta(u, v)$, of v from u (starting at time 0).*

Thus, no flow arrives to t (starting from s) on or before any time $l < \delta(s, t)$.

Definition 5 (Temporal Flow Network)**.** *A temporal flow network $\big(G(L), s, t, c, B\big)$ is a temporal graph $G(L) = (V, E, L)$ equipped with:*

1. *a source vertex s and a sink (target) vertex t,*
2. *for each edge e, a capacity $c_e > 0$; the capacities remain unchanged over time and are usually assumed to be integers,*
3. *for each node v, a buffer $B(v)$ of storage capacity (also referred to as* node capacity*) $B_v > 0$; node capacities also remain unchanged over time, and B_s and B_t are assumed to be infinite.*

We use c and B to denote the sets of edge and node capacities, respectively. If all node capacities are infinite, we denote the temporal flow network by $\big(G(L), s, t, c\big)$.

Definition 6 (Temporal Flows in Temporal Flow Networks)**.** *Let $\big(G(L) = (V, E, L), s, t, c, B\big)$ be a temporal flow network. Let:*

$$\begin{aligned} \mathcal{I}_u^+ &= \{e \in E | \exists w \in V, e = (u, w)\}, \\ \mathcal{I}_u^- &= \{e \in E | \exists w \in V, e = (w, u)\} \end{aligned}$$

be the outgoing and incoming edges to u. Also, let $L_R(u)$ be the set of labels on all edges incident to u along with an extra label 0 (artificial label for initialization), i.e.,

$$L_R(u) = \bigcup_{e \in \mathcal{I}_u^+ \cup \mathcal{I}_u^-} L_e \cup \{0\}.$$

A temporal flow on $G(L)$ consists of a non-negative real number $f(e, l)$ for each time edge (e, l), and real numbers $b_u^-(l), b_u^\mu(l), b_u^+(l)$ for each node $u \in V$ and each "day" l.

Note 1. *One may think of $b_v^-(l), b_v^\mu(l), b_v^+(l)$ as the buffer content of liquid in v at the "morning","noon", i.e., after the departures of flow from v, and "evening", i.e., after the arrivals of flow to v, of day l.*

The above numbers must satisfy all of the following:

1. $0 \leq f(e,l) \leq c_e$, *for every time edge* (e,l),
2. $0 \leq b_u^-(l) \leq B_u,\ 0 \leq b_u^\mu(l) \leq B_u,\ 0 \leq b_u^+(l) \leq B_u$, *for every node* u *and every* $l \in L_R(u)$,
3. *for every* $e \in E$, $f(e,0) = 0$,
4. *for every* $v \in V \setminus \{s\}$, $b_v^-(0) = b_v^\mu(0) = b_v^+(0) = 0$,
5. *for every* $e \in E$ *and* $l \notin L_e$, $f(e,l) = 0$,
6. *at time* 0 *there is a large enough amount of flow "units" available at the source* s *to allow as much flow as possible to move through the network,* $b_s^-(0) \geq \sum_{e \in \mathcal{I}_s^+} c_e |L_e|$,
7. *for every* $v \in V$ *and for every* $l \in L$, $b_v^-(l) = b_v^+(l_{prev})$, *where* l_{prev} *is the largest label in* $L_R(v)$ *that is smaller than* l,
8. *(Flow out on day* l*) for every* $v \in V$ *and for every* l, $b_v^\mu(l) = b_v^-(l) - \sum_{e \in \mathcal{I}_v^+} f(e,l)$,
9. *(Flow in on day* l*) for every* $v \in V$ *and for every* l, $b_v^+(l) = b_v^\mu(l) + \sum_{e \in \mathcal{I}_v^-} f(e,l)$.

Note 2. *The amount located at* s *at time* 0 *is chosen above to be at least* $\sum_{e \in \mathcal{I}_s^+} c_e |L_e|$. *Note that, no matter how much flow is located at* s *at time* 0, s *can only push* c_e *units of flow through every outgoing edge* e *at days of* e*'s availability. So even if more flow units were located at* s *at the start of time, the bound we have chosen above is an upper bound on the amount of flow that can travel through the whole network during its lifetime.*

Note 3. *For a temporal flow* f *on an acyclic* $G(L)$, *if one could guess the (real) numbers* $f(e,l)$ *for each time edge* (e,l), *then the numbers* $b_v^-(l), b_v^\mu(l), b_v^+(l)$, *for every* $v \in V$, *can be computed by a single pass over an order of the vertices of* $G(L)$ *from* s *to* t. *This can be done by following (1) through (9) from Definition 6 from* s *to* t.

Definition 7 (Value of a Temporal Flow)**.** *The value* $v(f)$ *of a temporal flow* f *is* $b_t^+(l_{max})$ *under* f, *i.e., the amount of liquid that, via* f, *reaches* t *during the lifetime of the network (*l_{max} *is the maximum label in* L*).*
If $b_t^+(l_{max}) > 0$ *for a particular flow* f, *we say that a portion of* f arrives *to* t.

Definition 8 (Mixed temporal networks)**.** *Given a directed graph* $G = (V,E)$ *with a source* s *and a sink* t *in* V, *let* $E = E_1 \cup E_2$, *so that* $E_1 \cap E_2 = \emptyset$, *and:*

1. *the labels (availabilities) of edges in* E_1 *are specified, and*
2. *each edge in* E_2 *receives a single label drawn uniformly at random from the set* $\{1, 2, \ldots, \alpha\}$, *for some even integer* α[2], *independently of the others.*

We call such a network "Mixed Temporal Network $[1,\alpha]$" *and denote it by* $G(E_1, E_2, \alpha)$.

The above definition only assigns a single random label to edges of E_2; however, one could also define mixed temporal networks where the edges could independently receive multiple random labels, each chosen uniformly at random from a set of available labels. Another alternative model of mixed temporal networks could consider edges that have both specified and randomly chosen labels. However, for the purpose of this work we focus only on the model of mixed temporal networks defined above, where each edge of E_2 receives a single random label.

Note that (traditional) temporal networks as previously defined are a special case of the mixed temporal networks, in which $E_2 = \emptyset$. However, with some edges being available at random times, the value of a temporal flow (until time α) becomes a random variable and the study of relevant problems requires a different approach than the one needed for (traditional) temporal networks.

Problem 1 (Maximum Temporal Flow (MTF))**.** *Given a temporal flow network* $\big(G(L), s, t, c, B\big)$ *and a day* $d \in \mathbb{N}$, *compute the maximum* $b_t^+(d)$ *over all flows* f *in the network.*

[2]We choose an even integer to simplify the calculations in the remainder of the paper. However, with careful adjustments, the results would still hold for an arbitrary integer.

2 Linear Program for the MTF problem with or without bounded buffers

In the description of the MTF problem, if d is not a label in L, it is enough to compute the maximum $b_t^+(l_m)$ over all flows, where l_m is the maximum label in L that is smaller than d. Henceforth, we assume $d = l_{max}$ unless otherwise specified; notice that the analysis does not change: if $d < l_{max}$, one can remove all time edges with labels larger than d and solve MTF in the resulting network with new maximum label at most d.

Note also that $b_t^+(l_{max})$ is not necessarily equal to the total outgoing flow from s during the lifetime of the network[3] , where the lifetime is $l_{max} - l_{min}$, l_{min} being the smallest label in the network. For example, consider the network of Figure 3. For $d = 5$, the *maximum flow by day* 5 *is* $b_t^+(5) = 8$, i.e., the flow where 5 units follow the journey $s \to v \to t$ and 3 units follow the journey $s \to u \to v \to t$; however, the total outgoing flow from s by day 5 is $10 > 8$.

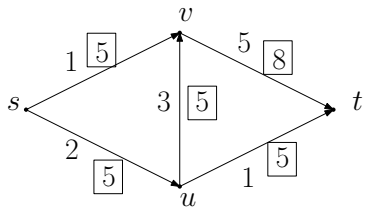


Figure 3: Outgoing flow from s is not always the same as maximum flow by some day d; here $d = 5$.

Let Σ be the set of conditions of Definition 6. The optimization problem, Π:

$$\left\{ \begin{array}{ll} \max \text{ (over all } f) & b_t^+(d) \\ \text{subject to} & \Sigma \end{array} \right\}$$

is a *linear program* with unknown variables $\{f(e,l), b_v^-(l), b_v^+(l), b_v^\mu(l)\}$, $\forall l \in L$, $\forall v \in V$, since each condition in Σ is either a linear equation or a linear inequality in the unknown variables. Notice that the number of equations and inequalities is polynomial in the size of the input of Π. This is obvious for the conditions 1, and 3 – 9 of Σ (Definition 6). For condition 2 of Σ, notice that there are $\Theta(|L_R(u)|)$ inequalities for every node u. $|L_R(u)|$ is the total number of labels assigned to edges incident to u. So, a very crude upper bound on the number of inequalities for every node is $(n-1) \cdot |E_L|$, where $|E_L|$ is the total number of labels assigned to the network. Thus, a crude upper bound on the total number of inequalities in condition 2 is $\Theta(n^2 \cdot |E_L|)$, which is polynomial in the input size.

Therefore, we get the following:

Lemma 1. *Maximum Temporal Flow is in P, i.e., can be solved in polynomial time in the size of the input, even when the node buffers are finite, i.e., bounded.*

Note 4. *Recall that E_L denotes the set of time edges of a temporal graph. If $n = |V|, m = |E|$ and $k = |E_L| = \sum_e |L_e|$, then MTF can be solved in sequential time polynomial in $n + m + k$ when the capacities and buffer sizes can be represented with polynomial in n number of bits. In the remainder of the paper, we shall investigate more efficient approaches for MTF.*

3 The time-extended flow network

Let $\big(G(L) = (V, E, L), s, t, c, B\big)$ be a temporal flow network on a directed graph G. Let E_L be the set of time edges of $G(L)$. Following the tradition in literature [24], we construct from $G(L)$ a *static* flow network called *the time-extended static flow network* that corresponds to $G(L)$, denoted by $\text{TEG}(L) = (V', E')$. By construction, $\text{TEG}(L)$ admits the same maximum flow as $G(L)$. $\text{TEG}(L)$ is constructed as follows.

[3]The total outgoing flow from s by some day x is the sum of all flow amounts that have "left" s by day x: $\sum_{l \in L_R(s) \setminus \{l^* \in \mathbb{N}: l^* > x\}} \sum_{e \in \mathcal{I}_s^+} f(e,l)$.

For every vertex $v \in V$, V' has a copy v_0 of v. V' also has a copy v_l of v if there is a time edge $(v, x, l) \in E_L$, for some $x \in V$, and a copy v_{l+tt} of v if there is a time edge $(x, v, l) \in E_L$, for some $x \in V$.

E' has a directed edge (called *vertical*) from a copy of vertex v to the *next* copy of v, for any $v \in V$. More specifically,

$$\forall v \in V,\ (v_i, v_j) \in E' \iff \begin{cases} v_i, v_j \in V', & \text{and} \\ j > i, & \text{and} \\ \forall k > i : v_k \in V' \implies k \geq j \end{cases}$$

Furthermore, for every time edge of $G(L)$, we consider a directed edge (called *crossing*) as follows:

$$\forall u, v \in V, l \in \mathbb{N},\ (u, v, l) \in E \iff (u_l, v_{l+tt}) \in E'$$

Every crossing edge $e \in \text{TEG}(L)$, i.e., every edge that connects copies of different vertices $u, v \in V$, has the capacity of the edge $(u, v) \in G(L)$, $c_e = c_{u,v}$. Every edge $e \in \text{TEG}(L)$ between copies of the same vertex $v \in V$ has capacity $c_e = B_v$. The source and target vertices in $\text{TEG}(L)$ are the first copy of s and the last copy of t in V' respectively. Note that $|V'| \leq |V| + 2|E_L|$ and $|E'| \leq |V| + 3|E_L|$.

Notice that $0 < tt < 1$ (by definition of the transit times), so if a vertex $v \in V$ has an incoming edge e with label l and an outgoing edge with label $l+1$, the copies v_{l+tt}, v_{l+1} of v in V' will never be identical (see Figure 4). This is important to ensure that $\text{TEG}(L)$ admits the same maximum flow as $G(L)$, for the case where node buffers are bounded. Indeed, assume that in the temporal network of Figure 4, the buffer of v has capacity $B_v = 5$ and the edge capacities are $c_{(u,v)} = c_{(v,w)} = 100$. Then the existence of the edge (v_{l+tt}, v_{l+1}) in $\text{TEG}(L)$ ensures that the corresponding (static) flow from v_l to w_{l+1+tt} is 5 (same as the 5 units of flow that we are allowed to store in $B(v)$ at any time), rather than 100 which would be the case if the copy v_{l+tt} of v was identical to the copy v_{l+1}.

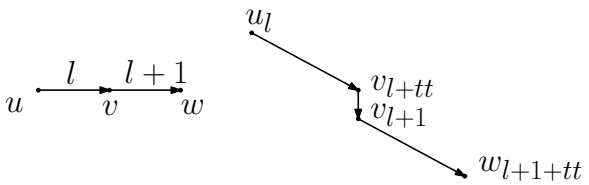


Figure 4: The copies of vertex v in $\text{TEG}(L)$.

Note 5. *If buffer capacities are infinite, one can omit vertices of the form v_{l+tt} and translate a time edge (u, v, l) of the temporal network to an edge (u_l, v_{l+1}) of* $\text{TEG}(L)$*. This would simplify the construction of* $\text{TEG}(L)$*, but cannot be done in general if node buffers are bounded.*

For $i = 1, 2, \ldots$, denote the i^{th} copy of any vertex $v \in V$ in the time-extended network by $v_{copy_{i-1}}$. Let also:

$$\begin{aligned} \mathcal{I}_u^+ &= \{e \in E | \exists w \in V, e = (u, w)\} \\ \mathcal{I}_u^- &= \{e \in E | \exists w \in V, e = (w, u)\} \end{aligned}$$

An $s \to t$ flow f in $G(L)$ defines an $s \to t$ flow (rate), f_R, in the time-extended network $\text{TEG}(L)$ as follows:

- The flow from the first copy of s to the next copy is the sum of all flow units that "leave" s in $G(L)$ throughout the time the network exists:

$$f_R(s_{copy_0}, s_{copy_1}) := \sum_{l \in \mathbb{N}} \sum_{e \in \mathcal{I}_s^+} f(e, l).$$

- The flow from the first copy of any *other* vertex to the next copy is zero:

$$\forall v \in V \setminus s,\ f_R(v_{copy_0}, v_{copy_1}) := 0.$$

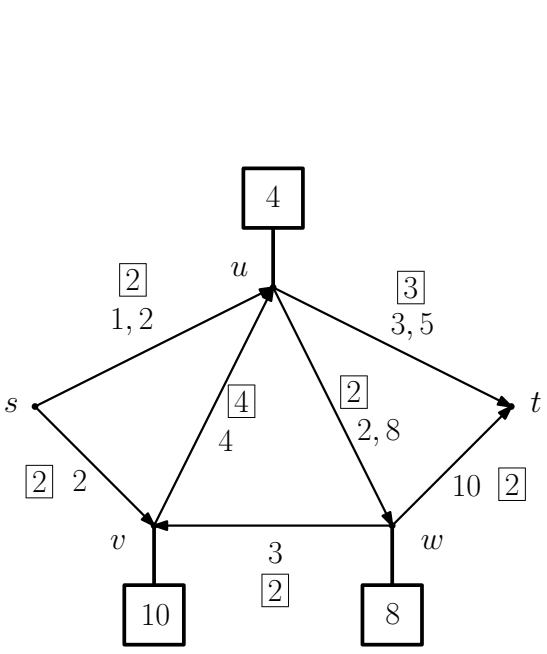
- The flow on any crossing edge that connects some copy u_l of vertex $u \in V$ and the copy v_{l+tt} of some other vertex $v \in V$ is exactly the flow on the time edge (u, v, l):

$$\forall (u_l, v_{l+tt}) \in E',\ f_R(u_l, v_{l+tt}) := f((u, v), l).$$

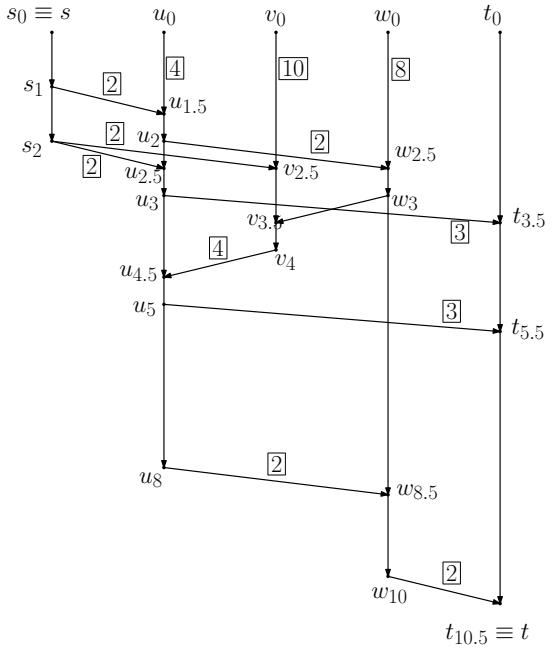
- The flow between two *consecutive* copies v_x and v_y, for some x, y, of the same vertex $v \in V$ corresponds to the units of flow stored in v from time x up to time y and is the difference between the flow *received* at the first copy through all incoming edges and the flow *sent* from the first copy through all outgoing crossing edges. So, $\forall v \in V, i = 1, 2, \ldots$, it is:

$$f_R(v_{copy_i}, v_{copy_{i+1}}) := \sum_{z \in V'} f_R(z, v_{copy_i}) - \sum_{u \in V' \setminus v_{copy_{i+1}}} f_R(v_{copy_i}, u).$$

Example Figure 5a shows a temporal network $G(L)$ with source s and sink t (node buffer capacities are in the boxes next to the nodes). The respective time-extended static graph TEG(L) is shown in Figure 5b. Edges between copies of s and between copies of t have infinite capacities (equal to the infinite capacity of the vertex buffer) which are not shown in the figure. For all other vertices, the capacities of edges between copies of the same vertex are drawn in a box only between the first and second copy due to lack of space in the figure, but are implied for all other edges between copies of the vertex.



(a) Temporal flow network $G(L)$



(b) Time extended network TEG(L)

Figure 5: Constructing the time-extended network

Let f_R be a static flow rate in the static network TEG(L) that corresponds to a temporal flow f in a temporal flow network $\big(G(L) = (V, E, L), s, t, c, B\big)$. By the construction of TEG(L), it follows:

Lemma 2. *Given a temporal flow network* $\big(G(L) = (V, E, L), s, t, c, B\big)$ *on a directed graph* G*,*

1. *The maximum temporal flow (from* s *to* t*),* $max_f v(f)$*, in* $G(L)$ *is equal to the maximum (standard) flow rate from* s *to* t *in the* static *network* TEG(L)*.*
2. *A temporal flow* f *is proper in* $G(L)$ *(i.e., satisfies all constraints) iff its corresponding static flow rate* f_R *is feasible in* TEG(L)*.*

It is also easy to see that:

Lemma 3. *Any static flow rate algorithm A that computes the maximum flow in a static, directed, s-t network G of n vertices and m edges in time $T(n, m)$, also computes the maximum temporal flow in a $\big(G(L) = (V, E, L), s, t, c, B\big)$ temporal flow network in time $T(n', m')$, where $n' \leq n + 2|E_L|$ and $m' \leq n + 3|E_L|$.*

Proof. We run A on the static network TEG(L) of n' vertices and m' edges. Note that TEG(L) is, by construction, acyclic. □

Note 6. *Our time-extended network has size (number of nodes and edges)* linear *on the input size of $G(L)$, and* not exponential.

The following is a direct corollary of the construction of the Time-Extended Graph and shows that any temporal flow from s to t can be decomposed into temporal flows on some $s \to t$ journeys.

Corollary 1 (Journeys flow decomposition)**.** *Let $\big(G(L) = (V, E, L), s, t, c, B\big)$ be a temporal flow network on a directed graph G. Let f be a temporal flow in $G(L)$ (f is given by the values of $f(e, l)$ for the time edges $(e, l) \in E_L$). Then, there is a collection of $s \to t$ journeys $j_1, j_2, \ldots, j_k$ such that:*

1. *$k \leq |E_L|$,*
2. *$v(f) = v(f_1) + \ldots v(f_k)$,*
3. *f_i sends positive flow only on the time edges of j_i.*

4 Temporal Networks with unbounded buffers at nodes

4.1 Maximum temporal flow – Minimum temporal cut

We consider here the MTF problem for temporal networks on underlying graphs with $B_v = +\infty$, $\forall v \in V$.

Definition 9 (Temporal Cut)**.** *Let $\big(G(L), s, t, c\big)$ be a temporal flow network on a digraph G. A set of time edges, S, is called a temporal cut (separating s and t) if the removal from the network of S results in a temporal flow network with* no $s \to t$ journey.

Definition 10 (Minimal Temporal Cut)**.** *A set of time edges, S, is called a minimal temporal cut (separating s and t) if:*

1. *S is a temporal cut, and*
2. *the removal from the network of any $S' \subset S$ results in a temporal flow network with at least one journey from s to t, i.e., no proper subset of S is a temporal cut.*

Definition 11. *Let S be a temporal cut of $\big(G(L) = (V, E, L), s, t, c\big)$. The* capacity *of the cut is $c(S) := \sum_{(e,l)\in S} c_e$.*

Consider the temporal network shown in Figure 6; here, a minimal temporal cut is $S = \{\big((s, v), 1\big), \big((s, v), 7\big)\}$ with capacity $c(S) = 20$. Notice that another minimal cut is $S' = \{\big((v, t), 8\big)\}$ with capacity $c(S') = 2$.

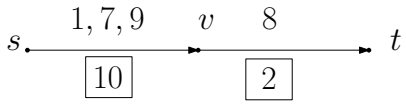


Figure 6: $S = \{\big((s, v), 1\big), \big((s, v), 7\big)\}$ is a minimal cut.

It follows from the definition of a temporal cut:

Lemma 4. *Let S be a (minimal) temporal cut in $\big(G(L) = (V, E, L), s, t, c\big)$. If we remove S from $G(L)$, no flow can ever arrive to t during the lifetime of $G(L)$.*

Proof. The removal of S leaves no $s \to t$ journey and any flow from s needs at least one journey to reach t, by definition. □

We are now ready to prove the main Theorem of this section:

Theorem 1. *The maximum temporal flow in* $\big(G(L) = (V, E, L), s, t, c\big)$ *is equal to the minimum capacity temporal cut.*

Proof. By Lemma 2, the maximum temporal flow in $G(L)$ is equal to the maximum flow rate from s to t in TEG(L). But in TEG(L), the maximum s-t flow rate is equal to the minimum s-t cut [24]. But any minimum capacity cut in TEG(L) only uses crossing edges and thus corresponds to a temporal cut in $G(L)$, of the same capacity (since the removal of the respective time edges leaves no $s \to t$ journey in $G(L)$). □

Open Problem. *The argument used above, that any minimum capacity cut in* TEG(L) *only uses crossing edges, cannot be extended to the case of temporal networks with bounded buffers. So, it remains open to:*

1. *give a meaningful definition of a temporal cut in the case of bounded buffers, and*
2. *possibly derive an analogue of the "maximum flow – minimum cut" Theorem for this case.*

4.2 Mixed Temporal Networks and their hardness

Mixed temporal networks of the form $G(E_1, E_2, \alpha)$ (see Definition 8) can model practical cases, where some edge availabilities are exactly specified, while some other edge availabilities are randomly chosen (due to security reasons, faults, etc.); for example, in a water network, one may have planned disruptions for maintenance in some water pipes, but unplanned (random) disruptions in some others. With some edges being available at random times, the value of the maximum temporal flow (until time α) now becomes a random variable.

In this section, we focus our attention to temporal networks that either have all their labels chosen uniformly at random, or are (fully) mixed.

4.2.1 Temporal Networks with random availabilities that are flow cutters

We study here a special case of the mixed temporal networks $G(E_1, E_2, \alpha)$, where $E_1 = \emptyset$, i.e., *all* the edges in the network become available at random time instances. We partially characterise such networks that eliminate the flow that arrives at t.

Let $G = (V, E)$ be a directed graph of n vertices with a distinguished source, s, and a distinguished sink, t. Suppose that each edge $e \in E$ is available only at a *unique* moment in time (i.e., day) *selected uniformly at random* from the set $\{1, 2, \ldots, \alpha\}$, for some even[4] integer $\alpha \geq 1$; suppose also that the selections of the edges' labels are independent. Let us call such a network a Temporal Network with unique random availabilities of edges, and denote it by URTN(α).

Lemma 5. *Let* P_k *be a directed* $s \to t$ *path of length* k *in* G*. Then,* P_k *becomes a journey in* URTN(α) *with probability at most* $\frac{1}{k!}$.

Proof. For a particular $s \to t$ path P_k of length k, let $\mathcal{E}$ be the event that "P_k is a journey", $\mathcal{D}$ be the event that "all k labels on P_k are different" and $\mathcal{S}$ be the event that "at least 2 out of the k labels on P_k are equal". Then, we have:

$$\begin{aligned} Pr[\mathcal{E}] &= Pr[\mathcal{E}|\mathcal{D}] \cdot Pr[\mathcal{D}] + Pr[\mathcal{E}|\mathcal{S}] \cdot Pr[\mathcal{S}] \\ &= Pr[\mathcal{E}|\mathcal{D}] \cdot Pr[\mathcal{D}] \\ &\leq Pr[\mathcal{E}|\mathcal{D}]. \end{aligned}$$

Now, each particular *set* of k different labels in the edges of P_k is equiprobable. But for each such set, all permutations of the k labels are equiprobable and only one is a journey, i.e., has increasing order of labels. Therefore:

$$Pr[P_k \text{ is a journey}] \leq \frac{1}{k!}.$$

□

[4]We choose an even integer to simplify the calculations. However, with careful adjustments to the calculations, the results would still hold for an arbitrary integer.

Definition 12 (c-long and thin graphs)**.** *A flow network is called a c-long and thin graph if the distance from s to t is at least $c \log n$, for a constant integer $c > 0$, and the number of simple directed $s \to t$ paths is at most n^β, for a constant β.*

Lemma 6. *Consider a $URTN(\alpha)$ with an underlying graph G being any particular c-long and thin digraph. Then, the probability that the amount of flow from s arriving at t is positive tends to zero as n tends to $+\infty$.*

Proof. The event E_1 ="at least one $s \to t$ path is a journey in $URTN(\alpha)$" is a prerequisite for a positive flow from s arriving at t. So,

$$\begin{aligned} Pr[\text{flow arriving at } t > 0] &= Pr[\exists s \to t \text{ simple path in } G \text{ which is a journey}] \\ &\leq n^\beta Pr[\text{any specific simple path in } G \text{ is a journey}] \\ &\leq n^\beta \frac{1}{(c \log n)!}, \end{aligned} \tag{1}$$

by Lemma 5 and since every $s \to t$ path in G has length at least $c \log n$. Since $(c \log n)!$ is superpolynomial and n^β is polynomial in n, we conclude that:

$$Pr[\text{flow arriving at } t > 0] \to 0, \text{ as } n \to +\infty.$$

□

Randomly labelled c-long and thin graphs are not the only class of temporal networks that disallows flow to arrive to t asymptotically almost surely.

Definition 13. *A cut C in a (traditional) flow network G is a set of edges, the removal of which from the network leaves no directed $s \to t$ paths in G.*

Definition 14. *A cut C_1* precedes *a cut C_2 in a flow network G (denoted by $C_1 \to C_2$) if any directed $s \to t$ path that goes through an edge in C_1 must also later go through an edge in C_2.*

Definition 15 (Multiblock graphs)**.** *A flow network is called a (c, d)-multiblock graph if it has at least $c \log n$ disjoint cuts $C_1, \ldots, C_{c \log n}$ such that $C_i \to C_{i+1}$, $i = 1, \ldots, c \log n - 1$, and for all $i = 1, \ldots, c \log n$, $|C_i| \leq d$, for some constants $c > 0$, $d > 2$.*

Note that (c, d)-multiblocks and (c-long,thin)-graphs are two different graph classes. Figure 7 shows a $(c, 2)$-multiblock of $n = ck + 2$, $k \in \mathbb{N}$, vertices which is not thin, i.e., the number of simple directed $s \to t$ paths is not polynomial to the number of its vertices.

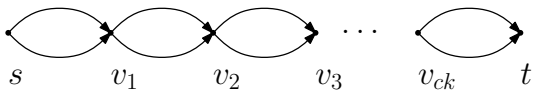


Figure 7: A $(c, 2)$-multiblock which is not thin.

Lemma 7. *Consider a $URTN(\alpha)$ with an underlying graph G being any particular (c, d)-multiblock. Then, the probability that the amount of flow from s arriving at t is positive tends to zero as n tends to $+\infty$.*

Proof. For positive flow to arrive to t starting from s, it must be that if $C_i \to C_{i+1}$ then at least one edge availability in C_{i+1} is larger than the smallest edge availability in C_i. Note that for every C_i, the probability that all labels in C_i are at least $\frac{\alpha}{2}$ is $\left(\frac{1}{2} + \frac{1}{\alpha}\right)^{|C_i|}$, i.e., a constant. Also, for every C_i, the probability that all labels in C_i are at most $\frac{\alpha}{2}$ is $\left(\frac{1}{2}\right)^{|C_i|}$, i.e., a constant.

Now, given a consecutive pair of cuts $C_i \to C_{i+1}$, let $E_{i,\geq}$ be the event that all labels in C_i are at least $\frac{\alpha}{2}$ and $E_{i+1,\leq}$ be the event that all labels in C_{i+1} are at most $\frac{\alpha}{2}$. Let A_i be the conjunction of $E_{i,\geq}$

and $E_{i+1,\leq}$. It holds that:

$$\begin{aligned} Pr[A_i] = Pr[E_{i,\geq} \wedge E_{i+1,\leq}] &= Pr[E_{i,\geq}] \cdot Pr[E_{i+1,\leq}] \\ &\geq \left(\frac{1}{2} + \frac{1}{\alpha}\right)^{|C_i|} \cdot \left(\frac{1}{2}\right)^{|C_{i+1}|} \\ &\geq \left(\frac{1}{2}\right)^{2d}. \end{aligned}$$

But, the conjunction of $E_{i,\geq}$ and $E_{i+1,\leq}$ implies that no flow arrives at t starting from s. Now, consider the events: $S = \{A_1, A_3, A_5, \ldots, A_r\}$, where r is the largest odd number that is smaller than $c \log n$; note that $r = \Theta(\log n)$. Those events are independent since there is no edge overlap in any of them; therefore, the random label choices in any one consecutive pair of cuts do not affect the choices in the next pair. We have:

$$\begin{aligned} Pr[\text{flow arriving at } t > 0] &\leq Pr[\text{all events in } S \text{ fail}] \\ &= \prod_{A_j \in S} Pr[A_j \text{ fails}] \\ &\leq \left(1 - \left(\frac{1}{2}\right)^{2d}\right)^{\Theta(\log n)} \xrightarrow{n \to +\infty} 0. \end{aligned}$$

This completes the proof of the Lemma. □

4.2.2 The complexity of computing the expected maximum temporal flow

We consider here the following problem:

Problem 2 (Expected Maximum Temporal Flow)**.** *Given a mixed temporal network $G(E_1, E_2, \alpha)$, compute the* expected value *of its maximum temporal flow, v.*

Let us recall the definition of the class of functions **#P**:

Definition 16. *[47, p.441] Let Q be a polynomially balanced, polynomial-time decidable binary relation. The* counting problem *associated with Q is: Given x, how many y are there such that $(x, y) \in Q$? **#P** is the class of all counting problems associated with polynomially balanced polynomial-time decidable functions.*

Loosely speaking, a problem is said to be **#P**-hard if a polynomial-time algorithm for it implies that **#P** = **FP**, where **FP** is the set of functions from $\{0,1\}^*$ to $\{0,1\}^*$ computable by a deterministic polynomial-time Turing machine[5]. For a more formal definition, see [47].

We now show the following:

Lemma 8. *Given an integer $C > 0$, it is **#P**-hard to compute the probability that the maximum flow value v in $G(E_1, E_2, \alpha)$ is at most C, $Pr[v \leq C]$.*

Proof. Recall that if $J = \{w_1, \ldots, w_n\}$ is a set of n positive integer weights and we are given an integer $C \geq \sum_{i=1}^{n} \frac{w_i}{2}$, then the problem of computing the number, T, of subsets of J with total weight at most C is **#P**-hard, because it is equivalent to counting the number of feasible solutions of the corresponding KNAPSACK instance [47].

Consider now the temporal flow network of Figure 8 where there are n directed disjoint two-edge paths from s to t. For the path with edges e_i, e'_i, via vertex v_i, the capacity of e_i is w_i and the capacity of e'_i is $w'_i \geq w_i$. In this network, $E_1 = \emptyset$ and $E_2 = E$, i.e., the availabilities of every edge are chosen independently and uniformly at random from $\{1, \ldots, \alpha\}$. For now let us just assume arbitrary positive integer α, and an appropriate choice of its value will be discussed later. Also, assume that each edge selects a *single* random label.

Clearly, the value of the maximum temporal flow from s to t until time $\alpha + tt$ is the sum of n random variables Y_i, $i = 1, \ldots, n$, where Y_i is the value of the flow through the i^{th} path. Y_i is, then, w_i with

[5] $\{0,1\}^* = \cup_{n \geq 0} \{0,1\}^n$, where $\{0,1\}^n$ is the set of all strings (of bits $0, 1$) of length n

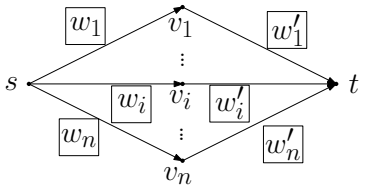


Figure 8: The network structure we consider

probability $p_i = \frac{1}{2} - \frac{1}{2\alpha}$, which is equal to the probability that the label l_{e_i} is smaller than the label $l_{e'_i}$, so that the path (e_i, e'_i) is a journey, and is zero otherwise. Then, $v = Y_1 + \ldots + Y_n$ and it holds that $Pr[v \leq C] = Pr[\sum_{i=1}^{n} Y_i \leq C]$.

Now, let J_k be the set of all vectors, $(\rho_1, \ldots, \rho_n)$, of n entries/weights in total, such that each ρ_i is either 0 or the corresponding w_i, and there are exactly k positive entries in the vector. Let $\vec{g} = (g_1, \ldots, g_n)$ be a specific assignment of weights to $Y_1, \ldots, Y_n$, respectively, i.e., $g_i = w_i$ with probability $\frac{1}{2} - \frac{1}{2\alpha}$ and, otherwise, $g_i = 0$; notice that $\vec{g} \in J_k$, for some $k \in \{0, \ldots, n\}$. Then,

$$\begin{aligned} Pr[v \leq C] &= Pr[\sum_{i=1}^{n} Y_i \leq C] \\ &= \sum_{\vec{g}} Pr[Y_i = g_i,\ \forall i = 1, \ldots, n] \cdot x(\vec{g}), \end{aligned} \tag{2}$$

where:

$$x(\vec{g}) = \begin{cases} 1 & \text{, if } \sum_{i=1}^{n} g_i \leq C \\ 0 & \text{, otherwise.} \end{cases}$$

For each particular $\vec{g}$ with exactly k positive weights, the probability that it occurs is $\left(\frac{1}{2} - \frac{1}{2\alpha}\right)^k \left(\frac{1}{2} + \frac{1}{2\alpha}\right)^{n-k}$. So, from Equation 2 we get:

$$\begin{aligned} Pr[v \leq C] &= \sum_{k=0}^{n} \sum_{\vec{g} \in J_k} x(\vec{g}) \left(\frac{1}{2} - \frac{1}{2\alpha}\right)^k \left(\frac{1}{2} + \frac{1}{2\alpha}\right)^{n-k} \\ &= \left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n \sum_{k=0}^{n} \sum_{\vec{g} \in J_k} x(\vec{g}) \left(\frac{\frac{1}{2} - \frac{1}{2\alpha}}{\frac{1}{2} + \frac{1}{2\alpha}}\right)^k. \end{aligned} \tag{3}$$

The following holds (using Bernoulli's inequality):

$$1 \geq \left(\frac{\frac{1}{2} - \frac{1}{2\alpha}}{\frac{1}{2} + \frac{1}{2\alpha}}\right)^k \geq \left(\frac{\frac{1}{2} - \frac{1}{2\alpha}}{\frac{1}{2} + \frac{1}{2\alpha}}\right)^n = \left(\frac{\alpha - 1}{\alpha + 1}\right)^n = \left(1 - \frac{2}{\alpha + 1}\right)^n \geq 1 - \frac{2n}{\alpha + 1}. \tag{4}$$

Let $T = \sum_{k=0}^{n} \sum_{\vec{g} \in J_k} x(\vec{g})$ and note that T is exactly the number of subsets of $J = \{w_1, \ldots, w_n\}$ with total weight at most C. Then, we get from Equation 3 and Relation 4:

$$\begin{array}{rcccl} \left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n \left(1 - \frac{2n}{\alpha+1}\right) T & \leq & Pr[v \leq C] & \leq & \left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n T \quad \Leftrightarrow \\ \left(1 - \frac{2n}{\alpha+1}\right) T & \leq & Pr[v \leq C] \frac{1}{\left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n} & \leq & T \quad \Leftrightarrow \\ T - \frac{2nT}{\alpha+1} & \leq & \frac{Pr[v \leq C]}{\left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n} & \leq & T. \end{array}$$

Now, assume that $\alpha + 1 > 2nT$; we can guarantee that by selecting α to be, for example, 2^n, or larger. Then, $0 < \frac{2nT}{\alpha+1} < 1$. Let $\varepsilon = \frac{2nT}{\alpha+1}$. Then, we get:

$$T - \varepsilon \leq \frac{Pr[v \leq C]}{\left(\frac{1}{2} + \frac{1}{2\alpha}\right)^n} \leq T.$$

Note that $\left(\frac{1}{2}+\frac{1}{2\alpha}\right)^n$ can be represented by a polynomial in n number of bits and can be computed in polynomial time.

If we had a polynomial-time algorithm, A, to exactly compute $Pr[v \leq C]$ for any C and α, then we could exactly compute (also in polynomial time) a number between $T-\varepsilon$ and T, for $0<\varepsilon<1$. But, this determines T exactly. So, such an algorithm A would solve a **#P**-hard problem in polynomial time. □

Remark 1. *If each of the random variables Y_i was of the form $Y_i = w_i$ with probability $p_i = \frac{1}{2}$, and zero otherwise, then the reduction from the KNAPSACK problem would be immediate [25, 37]. However, the possibility of ties in the various l_{e_i} and $l_{e'_i}$s excludes the respective journeys and the reduction does not carry out immediately.*

Now, given a mixed temporal network $G(E_1, E_2, \alpha)$, let v be the random variable representing the maximum temporal flow in G.

Definition 17. *The truncated by B expected maximum temporal flow of $G(E_1, E_2, \alpha)$, denoted by $E[v, B]$, is defined as:*

$$E[v, B] = \sum_{i=1}^{B} iPr[v = i].$$

Clearly, it is $E[v] = E[v, +\infty]$.

We are now ready to prove the main theorem of this section:

Theorem 2. *It is **#P**-hard to compute the expected maximum truncated Temporal Flow in a Mixed Temporal Network $G(E_1, E_2, \alpha)$.*

Proof. Consider the single-labelled mixed temporal network $G(E_1, E_2, \alpha)$ of Figure 9, in which s has n outgoing disjoint directed paths of two edges e_i, e'_i to a node t_1, and then there is an edge from t_1 to t. The capacity of each edge (s, v_i) , $i = 1, \ldots, n$, is w_i, the capacity of each edge (v_i, t_1) , $i = 1, \ldots, n$, is $w'_i \geq w_i$, and the capacity of the edge (t_1, t) is an integer B such that $\frac{1}{2}\sum_{i=1}^{n} w_i < B < \sum_{i=1}^{n} w_i$. The *unique label* of edge (t_1, t) is some $b \in \mathbb{N},\ b > \alpha$, where α is the maximum possible label that the other edges may select; in particular, each of the edges $(s, v_i), (v_i, t_1)$, $i = 1, \ldots, n$ receives a unique random label drawn uniformly and independently from $\{1, \ldots, \alpha\}$.

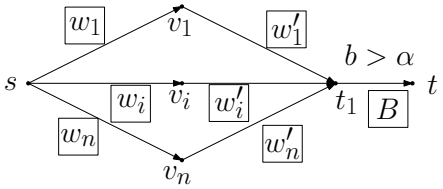


Figure 9: A $G(E_1, E_2, \alpha)$ where $E_1 = \{(t_1, t)\}$ with $l_{(t_1,t)} = b > \alpha$.

Clearly, the maximum temporal flow from s to t until time b is $v' = B$, if $v = \sum_{i=1}^{n} Y_i > B$, and is $v' = v = \sum_{i=1}^{n} Y_i$, otherwise; here $Y_i,\ i = 1, \ldots, n$, is the random variable representing the flow passing from t to t_1 via v_i in the time until α.

So, if $E[v']$ is the expected value of v', we have:

$$\begin{aligned} E[v'] &= \sum_{i=0}^{B} iPr[v = i] + B \cdot Pr[v > B] \\ &= E[v, B] + B\big(1 - Pr[v \leq B]\big). \end{aligned} \tag{5}$$

So, if we had a polynomial-time algorithm that could compute truncated expected maximum temporal flow values in mixed temporal networks, then we could compute $E[v']$ and $E[v, B]$; we could then solve Equation 5 for $Pr[v \leq B]$ and, thus, compute it in polynomial time. But to compute $Pr[v \leq B]$ is **#P**-hard by Lemma 8. □

Open Problem. *Although the computation of the expected maximum truncated temporal flow is **#P**-hard, it remains open whether computing the expected maximum temporal flow in mixed temporal networks is also hard.*

5 Conclusions

We defined and studied here for the first time flows in temporal networks. Our intuitive characterization of temporal cuts for networks with unbounded buffers may lead to fast algorithmic techniques (perhaps by sampling) for computing a minimum cut in such a network. We also considered random availabilities in some of the edges of our networks (mixed temporal networks). An interesting open problem is the existence of a FPTAS for the expected maximum flow value in mixed temporal networks, as well as a more thorough investigation of the complexity of its exact computation.

Further research on temporal flows in temporal networks could also consider *periodic* temporal graphs and the complexity of the maximum flow problem in them; these are graphs each edge e of which appears every x_e days ("edge period"). The maximum flow from s to t would then, in general, increase as we increase the day by which we wish to compute the flow that arrives at t. It seems that this problem requires a different approach than the one presented here, which also takes into account the different edge periods.

6 Acknowledgments

This work was supported in part by:

(i) the School of EEE and CS and the NeST initiative of the University of Liverpool,

(ii) the NSERC Discovery grant,

(iii) the EPSRC grant on "Algorithmic Aspects of Temporal Graphs" EP/P02002X/1, and

(iv) the AlgoUK network.

References

References

[1] Anna Adamaszek, Parinya Chalermsook, Alina Ene, and Andreas Wiese. Submodular unsplittable flow on trees. In *International Conference on Integer Programming and Combinatorial Optimization (IPCO)*, volume 9682 of *LNCS*, pages 337–349, 2016.

[2] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.

[3] Eleni C. Akrida, Leszek Gasieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.

[4] Eleni C. Akrida, Leszek Gąsieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: Diameter and connectivity. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 267–276, 2014.

[5] Eleni C. Akrida, Leszek Gąsieniec, George B. Mertzios, and Paul G. Spirakis. On temporally connected graphs of small cost. In *International Workshop on Approximation and Online Algorithms (WAOA)*, volume 9499 of *LNCS*, pages 84–96, 2015.

[6] Eleni C. Akrida and Paul G. Spirakis. On verifying and maintaining connectivity of interval temporal networks. In *International Symposium on Algorithms and Experiments for Wireless Sensor Networks (ALGOSENSORS)*, volume 9536 of *LNCS*, pages 142–154, 2015.

[7] Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\epsilon)$-approximate flow sparsifiers. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 279–293, 2014.

[8] J. E. Aronson. A survey of dynamic network flows. *Annals of Operations Research (Journal)*, 20:1–66, 1989.

[9] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5125 of *LNCS*, pages 121–132, 2008.

[10] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–58, 2015.

[11] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–58, 2015.

[12] Nadine Baumann and Martin Skutella. Earliest arrival flows with multiple sources. *Mathematics of Operations Research (Journal)*, 34(2):499–512, 2009.

[13] Kenneth A. Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks (Journal)*, 28(3):125–134, 1996.

[14] Natashia Boland, Thomas Kalinowski, Hamish Waterer, and Lanbo Zheng. Scheduling arc maintenance jobs in a network to maximize total flow over time. *Discrete Applied Mathematics (Journal)*, 163:34–52, 2014.

[15] Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Formal Models and Metrics . *Tech. Rep. Archive ouverte HAL,*, 2013.

[16] Arnaud Casteigts and Paola Flocchini. Deterministic Algorithms in Dynamic Networks: Problems, Analysis, and Algorithmic Tools. *Tech. Rep. Archive ouverte HAL,*, 2013.

[17] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[18] Augustin Chaintreau, Abderrahmen Mtibaa, Laurent Massoulié, and Christophe Diot. The diameter of opportunistic mobile networks. In *ACM Conference on Emerging Network Experiment and Technology (CoNEXT)*, pages 1–12, 2007.

[19] Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics*, 24(4):1694–1712, 2010.

[20] Friedrich Eisenbrand, Andreas Karrenbauer, Martin Skutella, and Chihao Xu. Multiline addressing by network flow. *Algorithmica (Journal)*, 53(4):583–596, 2009.

[21] Khaled M. Elbassioni, Naveen Garg, Divya Gupta, Amit Kumar, Vishal Narula, and Arindam Pal. Approximation algorithms for the unsplittable flow problem on paths and trees. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 267–275, 2012.

[22] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134 of *LNCS*, pages 444–455, 2015.

[23] Lisa Fleischer and Martin Skutella. Quickest flows over time. *SIAM Journal on Computing*, 36(6):1600–1630, 2007.

[24] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 2010.

[25] Dimitris Fotakis, Spyros C. Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul G. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theoretical Computer Science (Journal)*, 410(36):3305–3326, 2009.

[26] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow: Extended abstract. In *ACM Symposium on Principles of Distributed Computing (PODC)*, pages 81–90, 2015.

[27] Martin Groß and Martin Skutella. Generalized maximum flows over time. In *International Workshop on Approximation and Online Algorithms (WAOA)*, volume 7164 of *LNCS*, pages 247–260, 2011.

[28] Mohammad Taghi Hajiaghayi and Harald Räcke. An o(sqrt(n))-approximation algorithm for directed sparsest cut. *Information Processing Letters (Journal)*, 97(4):156–160, 2006.

[29] Horst W. Hamacher and Stevanus A. Tjandra. Earliest arrival flows with time-dependent data. Tech. Rep. in Wirtschaftsmathematik (WIMA Report) 88, Fachbereich Mathematik `https://kluedo.ub.uni-kl.de/frontdoor/index/index/docId/1449`, 2003.

[30] Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Adapting the bron-kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining (Journal)*, 7(1):35:1–35:16, 2017.

[31] Bruce Hoppe and Éva Tardos. The quickest transshipment problem. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 512–521, 1995.

[32] Bruce Hoppe and Éva Tardos. The quickest transshipment problem. *Mathematics of Operations Research (Journal)*, 25(1):36–62, 2000.

[33] Bruce Edward Hoppe. *PhD thesis: Efficient Dynamic Network Flow Algorithms*. Cornell University, 1995.

[34] Naoyuki Kamiyama and Naoki Katoh. The universally quickest transshipment problem in a certain class of dynamic networks with uniform path-lengths. *Discrete Applied Mathematics (Journal)*, 178:89–100, 2014.

[35] Jan-Philipp W. Kappmeier. *Generalizations of Flows over Time with Applications in Evacuation Optimization*. epubli GmbH, 2015.

[36] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.

[37] Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM Journal on Computing*, 30(1):191–217, 2000.

[38] Bettina Klinz and Gerhard J. Woeginger. One, two, three, many, or: complexity aspects of dynamic network flows with dedicated arcs. *Operations Research Letters (Journal)*, 22(4-5):119–127, 1998.

[39] Ronald Koch, Ebrahim Nasrabadi, and Martin Skutella. Continuous and discrete flows over time - A general model based on measure theory. *Mathematical Methods of Operations Research (Journal)*, 73(3):301–337, 2011.

[40] Jakub Lacki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 599–608, 2012.

[41] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 245–254, 2010.

[42] George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 7966 of *LNCS*, pages 657–668, 2013.

[43] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. In *International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 8635 of *LNCS*, pages 553–564, 2014.

[44] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal Networks*, Understanding Complex Systems, pages 15–40. Springer, 2013.

[45] Regina O'Dell and Roger Wattenhofer. Information dissemination in highly dynamic graphs. In *Joint workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 104–110, 2005.

[46] James B. Orlin. Max flows in o(nm) time, or better. In *ACM Symposium on Theory of Computing (STOC)*, pages 765–774, 2013.

[47] Christos M. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[48] Warren B Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. *Handbooks in operations research and management science*, 8:141–295, 1995.

[49] Tomasz Radzik. Minimizing capacity violations in a transshipment network. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 185–194, 1992.

[50] Tomasz Radzik. Faster algorithms for the generalized network flow problem. *Mathematics of Operations Research (Journal)*, 23(1):69–100, 1998.

[51] Tomasz Radzik. Improving time bounds on maximum generalised flow computations by contracting the network. *Theoretical Computer Science (Journal)*, 312(1):75–97, 2004.

[52] Christian Scheideler. Models and techniques for communication in dynamic networks. In *Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285 of *LNCS*, pages 27–49, 2002.

[53] Maria J. Serna. Randomized parallel approximations to max flow. In *Encyclopedia of Algorithms*, pages 1750–1753. Springer, 2016.

[54] Martin Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization*, pages 451–482, 2008.